An Introduction

Frederik Fiand & Tim Johannessen

GAMS Software GmbH

Agenda

- **GAMS at a Glance**
- GAMS - Hands On Examples
- APIs - Application Programming Interfaces to GAMS
- Outlook - Some Advanced GAMS Features

Company

- Roots: World Bank, 1976
- Went commercial in 1987
- Locations
  - GAMS Development Corporation (Washington)
  - GAMS Software GmbH (Germany)
- Product: The **G**eneral **A**lgebraic **M**odeling **S**ystem

GAMS

What did this give us?

Simplified model development & maintenance

Increased productivity tremendously

Made mathematical optimization available to a broader audience (domain experts)

➢ 2012 INFORMS Impact Prize

Broad User Community and Network

- 11,500+ licenses
- Users: 50% academic, 50% commercial
- GAMS used in more than 120 countries
- Uniform interface to more than 30 solvers

25+ Years
GAMS Development

GAMS

Broad Range of Application Areas

Agricultural Economics	Applied General Equilibrium
Chemical Engineering	Economic Development
Econometrics	Energy
Environmental Economics	Engineering
Finance	Forestry
International Trade	Logistics
Macro Economics	Military
Management Science/OR	Mathematics
Micro Economics	Physics

25+ Years
GAMS Development

Foundation of GAMS

Powerful algebraic modeling language

Open architecture and interfaces to other systems, independent layers

Powerful Declarative Language

Similar to mathematical notation

Easy to learn - few basic language elements: sets, parameters, variables, equations, models

Model is executable (algebraic) description of the problem

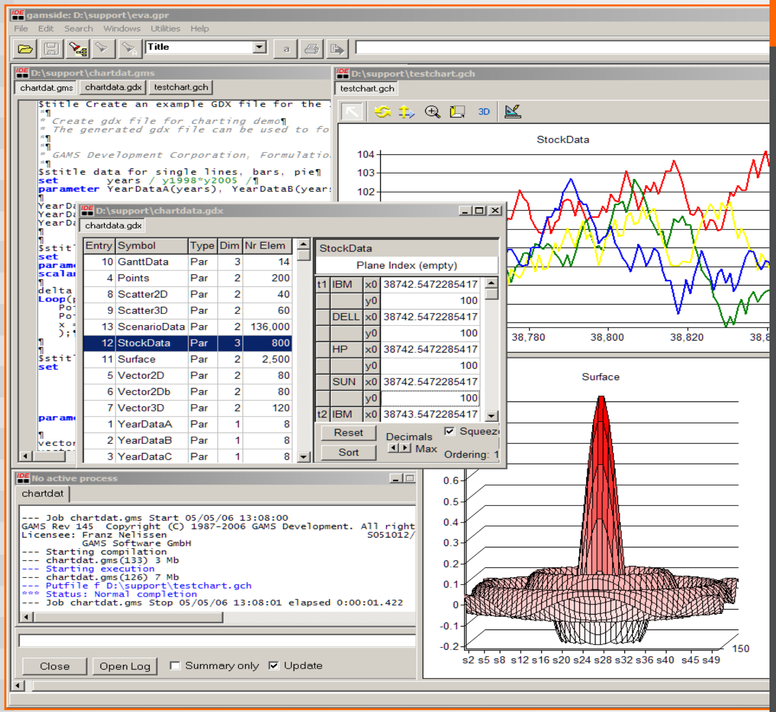
Mix of Declarative and Imperative Elements

Control Flow Statements (e.g. loops, for, if,…), macros and functions

Advantages:

- Build complex problem algorithms within GAMS
- Simplified interaction with other systems:
  - Data exchange
  - GAMS process control

Strong Development Environment



GAMS IDE

- Project management
- Editor / Syntax coloring / Spell checks
- Tree view / Syntax-error navigation
- Model Debugging & Profiling
- Solver selection & setup
- Data viewer
  - Export
  - Charting
- GAMS Processes Control

Independence of Model and Operating System

Microsoft Windows

Linux

Welcome to Mac OS X

solaris

AIX

Platforms supported by GAMS:

➡ Models can be moved between platforms with ease!

Model

Platform | Solver | Data | Interface

Independence of Model and Solver

One environment for a wide range of model types and solvers

- All major commercial LP/MIP solver
- Open Source Solver (COIN)
- Also solver for NLP, MINLP, global, and stochastic optimization

FICO | Gurobi Optimization | IBM | mosek | ... | COIN|OR

➡ Switching between solvers with one line of code!

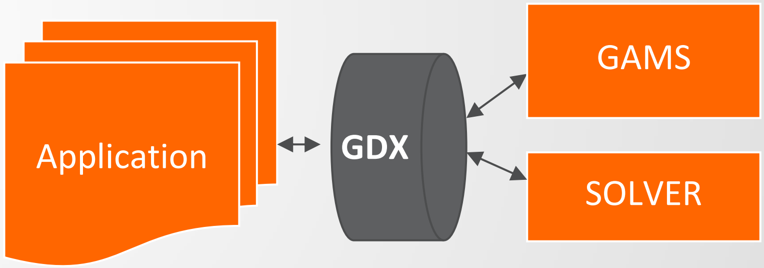
Model			
Platform	Solver	Data	Interface

GAMS

Independence of Model and Data

- Declarative Modeling
- ASCII: Initial model development

- GDX: Data layer ("contract") between GAMS and applications
  - Platform independent
  - No license required
  - Direct GDX interfaces and general API
  - …



Model

Platform | Solver | Data | Interface

GAMS

Independence of Model and User Interface

API's

- *Low Level*
- **Object Oriented**: .Net, Java, Python
- No modeling capability: Model is written in GAMS
- Wrapper class that encapsulates a GAMS model

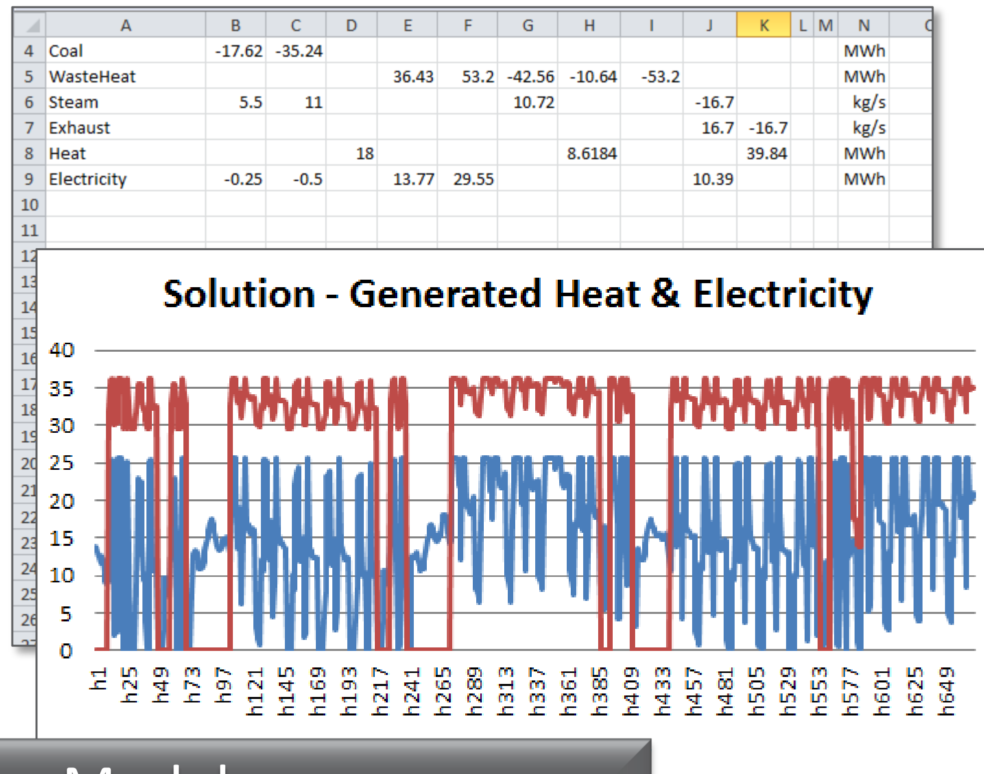
Model			
Platform	Solver	Data	Interface

GAMS

Smart Links to other Applications

- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application

MS Excel



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
4	Coal	-17.62	-35.24											MWh
5	WasteHeat				36.43	53.2	-42.56	-10.64	-53.2					MWh
6	Steam	5.5	11				10.72			-16.7				kg/s
7	Exhaust									16.7	-16.7			kg/s
8	Heat			18				8.6184			39.84			MWh
9	Electricity	-0.25	-0.5		13.77	29.55				10.39				MWh

Model

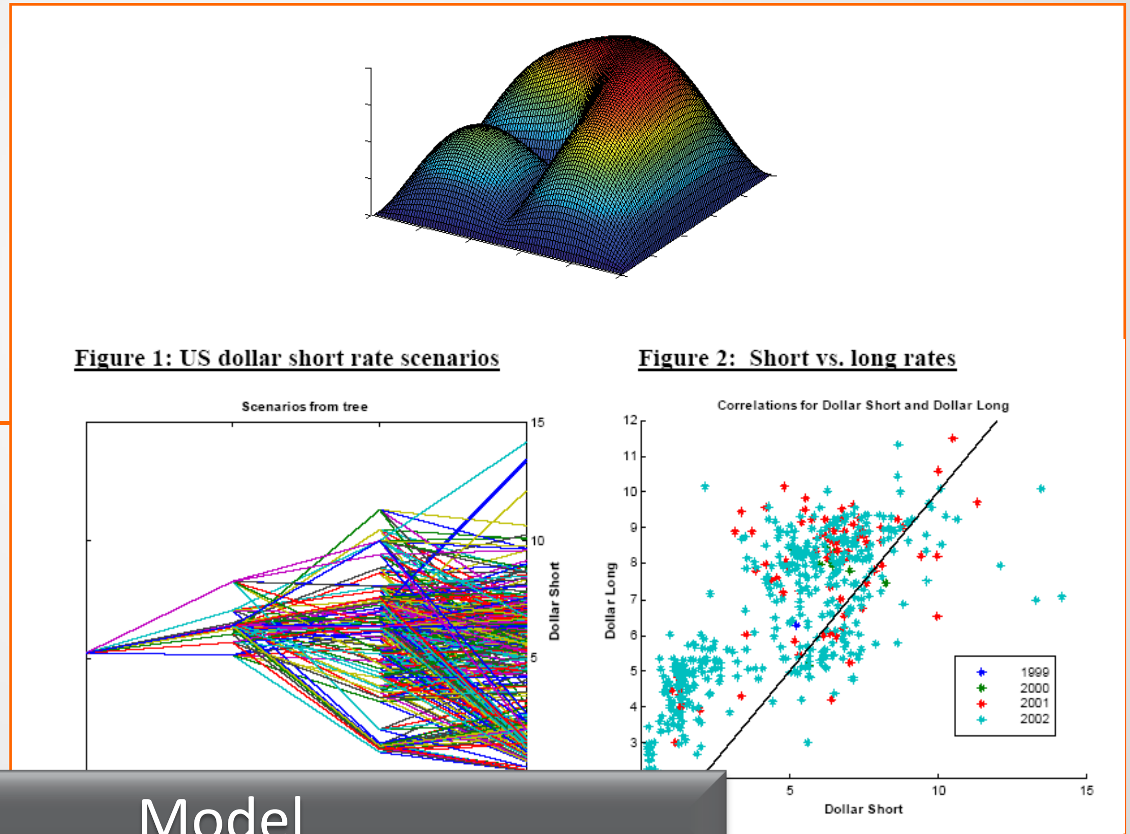
Platform | Solver | Data | Interface

GAMS

Smart Links to other Applications

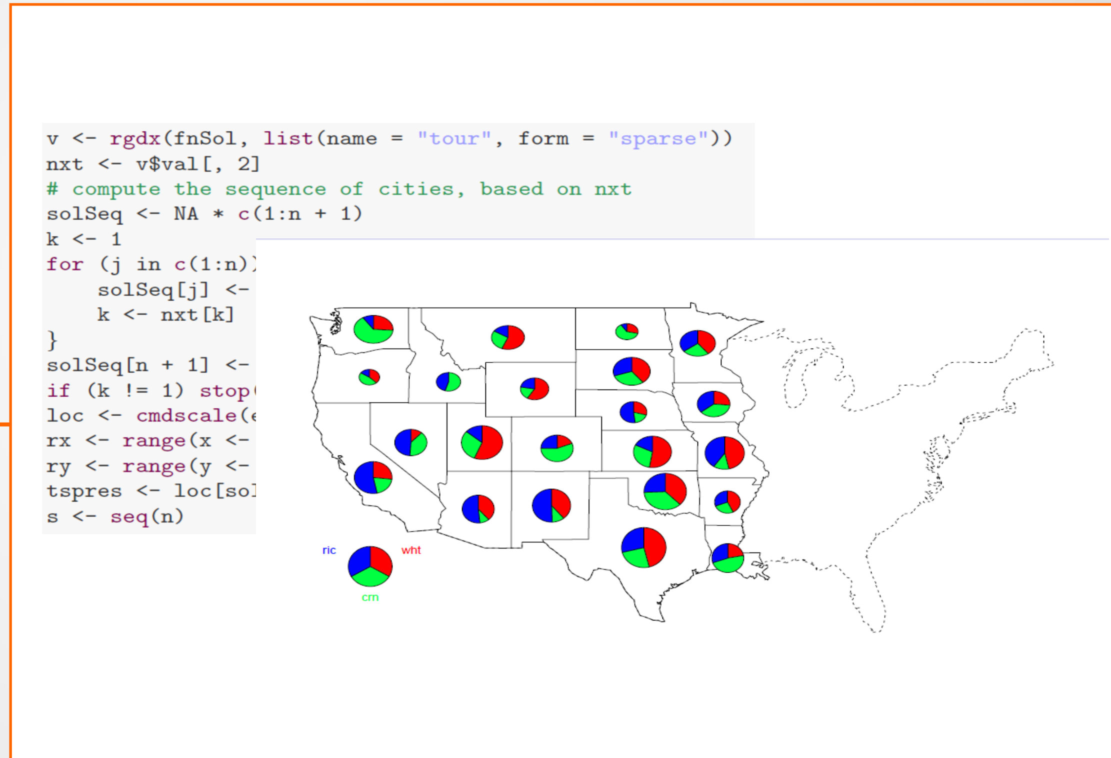
- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application

MatLab



Smart Links to other Applications

- User keeps working in his productive tool environment
- Application accesses all optimization capabilities of GAMS through API
- Visualization and analysis of model data and results in the application



Striving for Innovation and Compatibility

Models must benefit from:

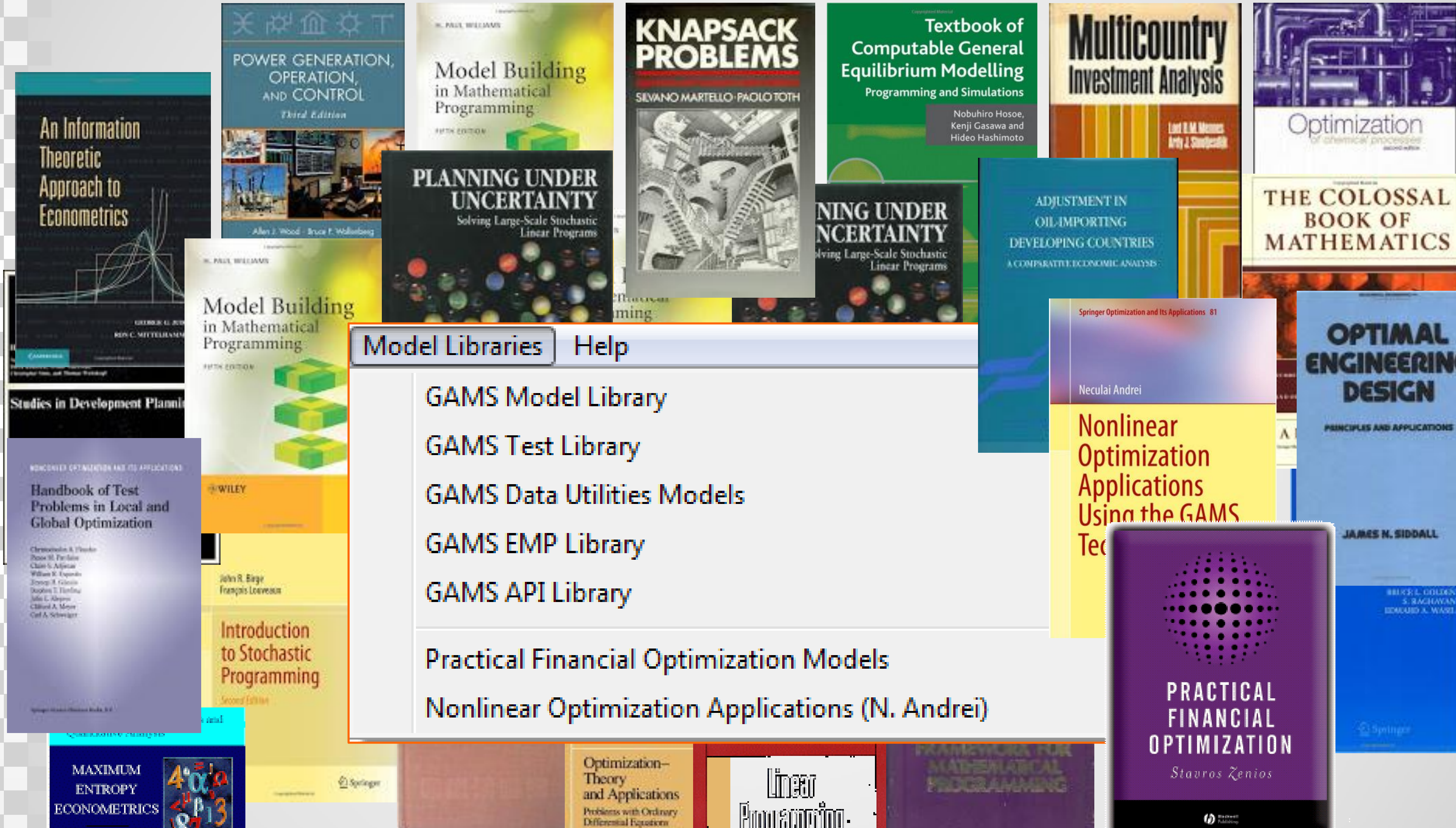
- Advancing hardware / New Platforms
- Enhanced / new solver and solution technology
- Improved / upcoming interfaces to other systems
- **New Modeling Concepts**

Protect investments of Users

- Life time of a model: 15+ years
- New maintainer, platform, solver, user interface
- Backward Compatibility
- **Software Quality Assurance**

GAMS

Free Model Libraries



➢ More than 1400 models!

GAMS

Why GAMS?

- Experience of 25+ years
- Broad user community from different areas
- Lots of model templates
- Strong development interface

- Consistent implemenation of design principles
  - Simple, but powerful modeling language
  - Independent layers
  - Open architecture: Designed to interact with other applications

- Open for new developments
- Protecting investments of users

Agenda

- GAMS at a Glance
- **GAMS - Hands On Examples**
- APIs - Application Programming Interfaces to GAMS
- Outlook - Some Advanced GAMS Features

GAMS

A Simple Transportation Problem

What does this example show?

- It gives a first glimpse of how a problem can be formulated in GAMS
- It shows some basics of data exchange with GAMS
- It shows how easy it is to change model type and, consequently, solver technology

Model types in this example

LP

- Determine minimum transportation cost.
  Result: city to city shipment volumes.

MIP

- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.

MINLP

- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

SP

- Allows uncertainty,
  e.g. uncertain demand

Model types in this example

LP

- Determine minimum transportation cost.
  Result: city to city shipment volumes.

MIP

- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.

MINLP

- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

SP

- Allows uncertainty,
  e.g. uncertain demand

A Simple Transportation Problem

Canning Plants (supply) → shipments (Number of cases) → **Markets** (demand)

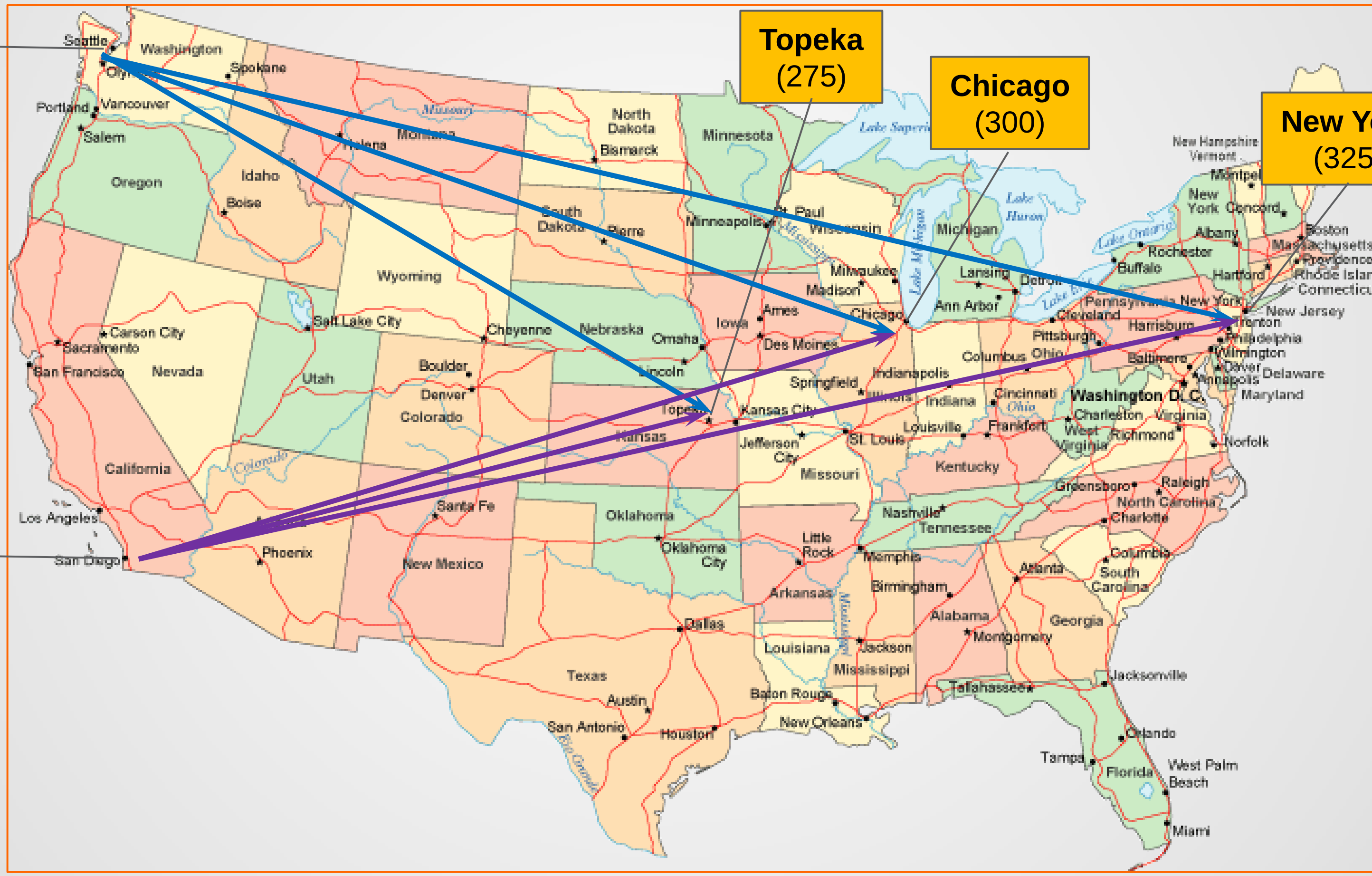
Seattle (350)

San Diego (600)

Topeka (275)

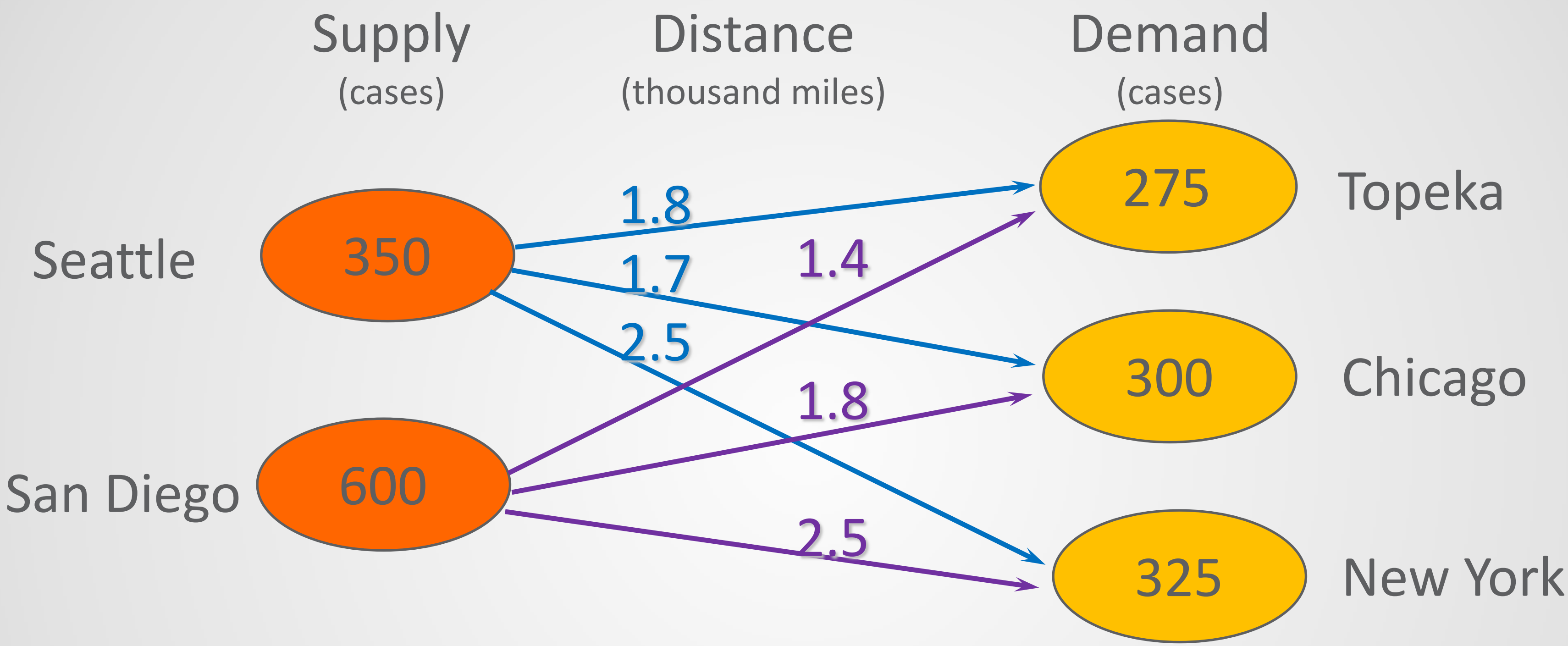
Chicago (300)

New York (325)



Freight: $90 case / thousand miles

A Simple Transportation Problem



Freight: $90 case / thousand miles

Minimize	Transportation cost
subject to	Demand satisfaction at markets
	Supply constraints

GAMS

Mathematical Model Formulation

Indices: i (Canning plants)

j (Markets)

Decision variables: x_{ij} (Number of cases to ship)

Data: c_{ij} (Transport cost per case)

a_i (Capacity in cases)

b_i (Demand in cases)

$\min \quad \sum_i \sum_j c_{ij} \cdot x_{ij}$ (Minimize total transportation cost)

subject to

$\sum_j x_{ij} \leq a_i \quad \forall\, i$ (Shipments from each plant $\leq$ supply capacity)

$\sum_i x_{ij} \geq b_j \quad \forall\, j$ (Shipments to each market $\geq$ demand)

$x_{ij} \geq 0 \quad \forall i,j$ (Do not ship from market to plant)

$i,j \in \mathbb{N}$

GAMS Syntax (LP Model)

gamside: C:\Users\ToniLapTop\Documents\gamsdir\projdir\Transport\tmp.gpr - [C:\Users\ToniLapTop\...

trnsport_LP_MIP_MINLP.gms

```
Variables
     x(i,j)          shipment quantities in cases
     z               total transportation costs in thousands of dollars ;

Positive Variable x ;

Equations
     cost            define objective function
     supply(i)       observe supply limit at plant i
     demand(j)       satisfy demand at market j ;

cost ..              z  =e=  sum((i,j), c(i,j)*x(i,j)) ;

supply(i) ..         sum(j, x(i,j))  =l=  a(i) ;

demand(j) ..         sum(i, x(i,j))  =g=  b(j) ;

Model modelLP   /cost, supply, demand/ ;

Solve modelLP using lp minimizing z ;
```

GAMS Syntax (LP Model)

gamside: C:\Users\ToniLapTop\Documents\gamsdir\projdir\Transport\tmp.gpr - [C:\Users\ToniLapTop\...

File Edit Search Windows Utilities Model Libraries Help

trnsport_LP_MIP_MINLP.gms

```
Variables
     x(i,j)         shipment quantities in cases
     z              total transportation costs in thousands of dollars ;

Positive Variable x ;

Equations
     cost           define objective function
     supply(i)      observe supply limit at plant i
     demand(j)      satisfy demand at market j ;

cost ..             z  =e=  sum((i,j), c(i,j)*x(i,j)) ;

supply(i) ..        sum(j, x(i,j))  =l=  a(i) ;

demand(j) ..        sum(i, x(i,j))  =g=  b(j) ;

Model modelLP   /cost, supply, demand/ ;

Solve modelLP using lp minimizing z ;
```

1: 1 Insert

Hands-On

GAMS

GAMS Syntax (Data Input)

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

optfile

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\data.gms

data.gms

```
Sets
     i   canning plants   / seattle, san-diego /
     j   markets          / new-york, chicago, topeka / ;

Parameters

     a(i)  capacity of plant i in cases
       /    seattle     350
            san-diego   600  /

     b(j)  demand at market j in cases
       /    new-york    325
            chicago     300
            topeka      275  / ;

Table d(i,j)  distance in thousands of miles
                  new-york       chicago      topeka
    seattle          2.5           1.7          1.8
    san-diego        2.5           1.8          1.4  ;

Scalar f  freight in dollars per case

Parameter c(i,j)  transport cost in t

          c(i,j) = f * d(i,j) / 1000
```

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconfere...

2_trnsport_include_data.gms

```
$include data.gms

 Variables
      x(i,j)  shipment quantities in cases
      z       total transportation costs in thousa

 Positive Variable x ;

 Equations
      cost        define objective function
      supply(i)   observe supply limit at plant i
      demand(j)   satisfy demand at market j ;
```

GAMS Syntax (Data Input)

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

optfile

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\3_trnsport_gdx.gms

3_trnsport_gdx.gms | data.gms

```
  Sets
       i   canning plants
       j   markets        ;

  Parameters
       a(i)   capacity of plant i in cases
       b(j)   demand at market j in cases
       d(i,j) distance in thousands of miles
       c(i,j) transport cost in thousands of dollars per case ;

$call gams data.gms gdx=data.gdx
$if errorlevel 1 $abort Error preparing data
$gdxin data.gdx
$load i j a b d
$gdxin

  Scalar f  freight in dollars per case per thousand miles  /90/ ;
  c(i,j) = f * d(i,j) / 1000 ;

  Variables
       x(i,j)  shipment quantities in cases
       z       total transportation costs in thousands of dollars ;

  Positive Variable x ;

  Equations
       cost        define objective function
       supply(i)   observe supply limit at plant i
       demand(j)   satisfy demand at market j ;
```

GAMS Syntax (Data Input)

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\4_trnsport_gdxxrw.gms

4_trnsport_gdxxrw.gms

```
 Sets
      i   canning plants
      j   markets        ;

 Parameters
      a(i)   capacity of plant i in cases
      b(j)   demand at market j in cases
      d(i,j) distance in thousands of miles
      c(i,j) transport cost in thousands of dollars per case ;



$onecho > instructions.txt
par=d rng=Sheet1!A1 rdim=1 cdim=1
par=b rng=Sheet1!B6 rdim=0 cdim=1
par=a rng=Sheet1!G2 rdim=1 cdim=0
$offecho
$call gdxxrw data.xlsx @instructions.txt
$if errorlevel 1 $abort Error preparing data
$gdxin data.gdx
$load i<d.dim1 j<d.dim2 d a b
$gdxin

 Scalar f  freight in dollars per case per thousand miles  /90/ ;
 c(i,j) = f * d(i,j) / 1000 ;

 Variables
      x(i,j)   shipment quantities in cases
      z        total transportation costs in thousands of dollars ;
```

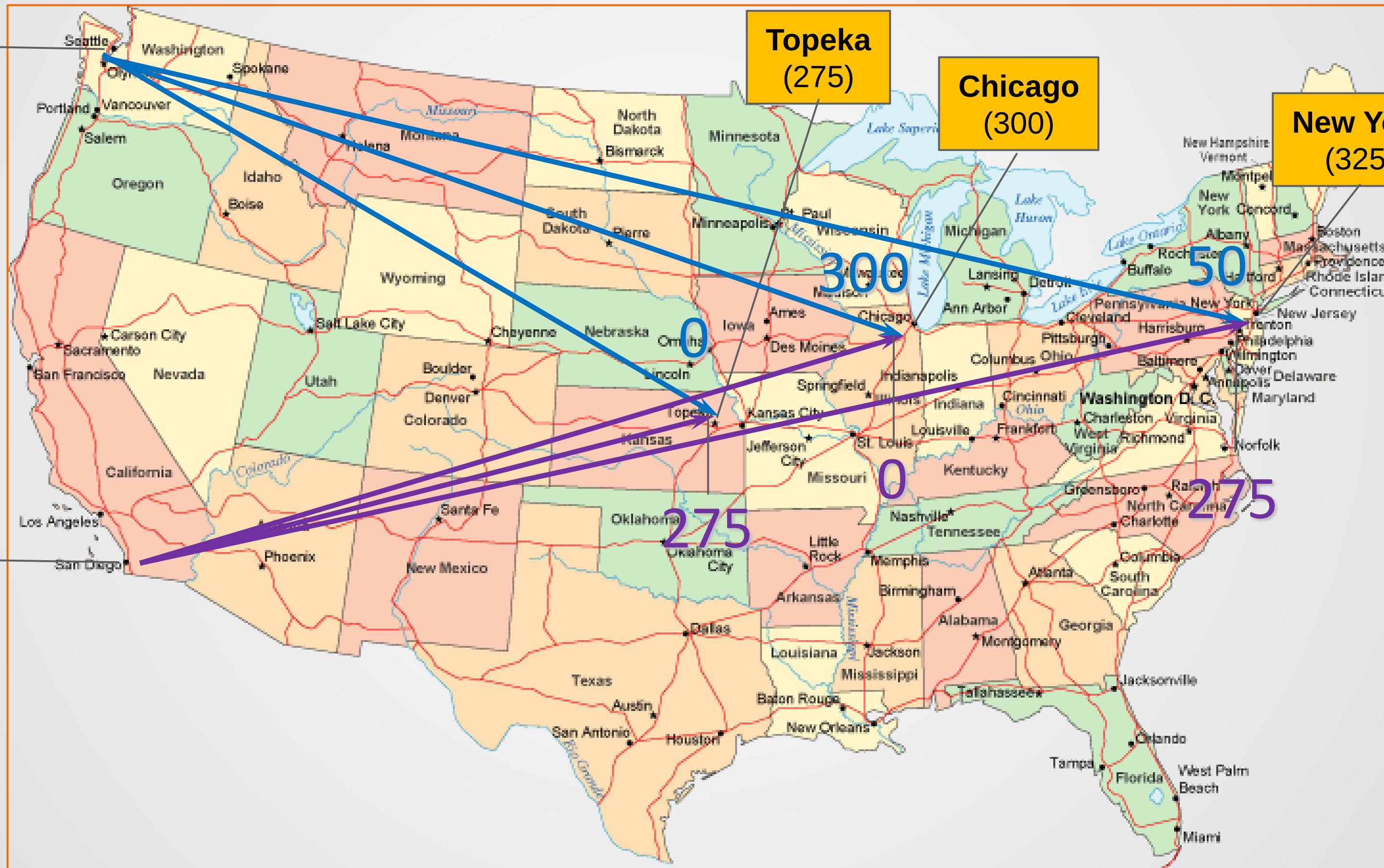
GAMS

Solution to LP model

Canning Plants (supply) — shipments → (Number of cases) — **Markets** (demand)

Seattle (350)

San Diego (600)



Freight: $90 case / thousand miles

Total cost: $153,675

Model types in this example

LP

- Determine minimum transportation cost.
  Result: city to city shipment volumes.

↓

MIP

- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.

↓

MINLP

- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

↓

SP

- Allows uncertainty,
  e.g. uncertain demand

Model types in this example

LP

- Determine minimum transportation cost.
  Result: city to city shipment volumes.

MIP

- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.

MINLP

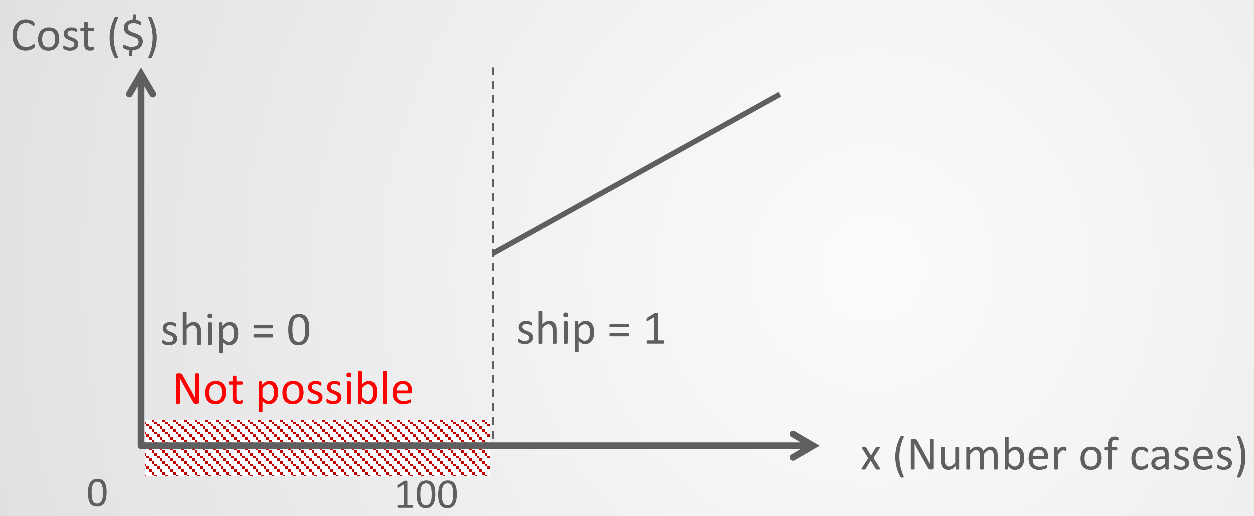
- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

SP

- Allows uncertainty,
  e.g. uncertain demand

MIP Model: Minimum Shipment of 100 cases

- Shipment volume: **x** (continuous variable)
- Discrete decision: **ship** (binary variable)



add constraints:

$x_{i,j} \geq 100 \cdot ship_{i,j} \quad \forall\, i,j$ (if ship=1, then ship at least 100)

$x_{i,j} \leq bigM \cdot ship_{i,j} \quad \forall\, i,j$ (if ship=0, then do not ship at all)

$ship_{i,j} \in \{0,1\}$

GAMS

MIP Model: GAMS Syntax

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

gdx results

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\5_trnsport_MIP.gms

5_trnsport_MIP.gms 5_trnsport_MIP.lst

```
  demand(j) ..   sum(i, x(i,j))  =g=  b(j) ;

  model transportLP / all /;

  Solve transportLP using LP minimizing z ;

parameter rep(i,j,*) report parameter;
rep(i,j,'LP') = x.l(i,j);

* MIP
scalar minS minimum shipment / 100 /
       bigM big M;
bigM = min(smax(i,a(i)), smax(j,b(j)));

binary variable ship(i,j) '1 if we ship from i to j, otherwise 0';

equation minship(i,j) minimum shipment
         maxship(i,j) maximum shipment;

minship(i,j).. x(i,j) =g= minS * ship(i,j);
maxship(i,j).. x(i,j) =l= bigM * ship(i,j);

Model transportMIP / transportLP, minship, maxship / ;
option optcr = 0;

Solve transportMIP using MIP minimizing z ;

rep(i,j,'MIP') = x.l(i,j);
display rep;
```

MIP Model: GAMS Syntax

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

gdx results

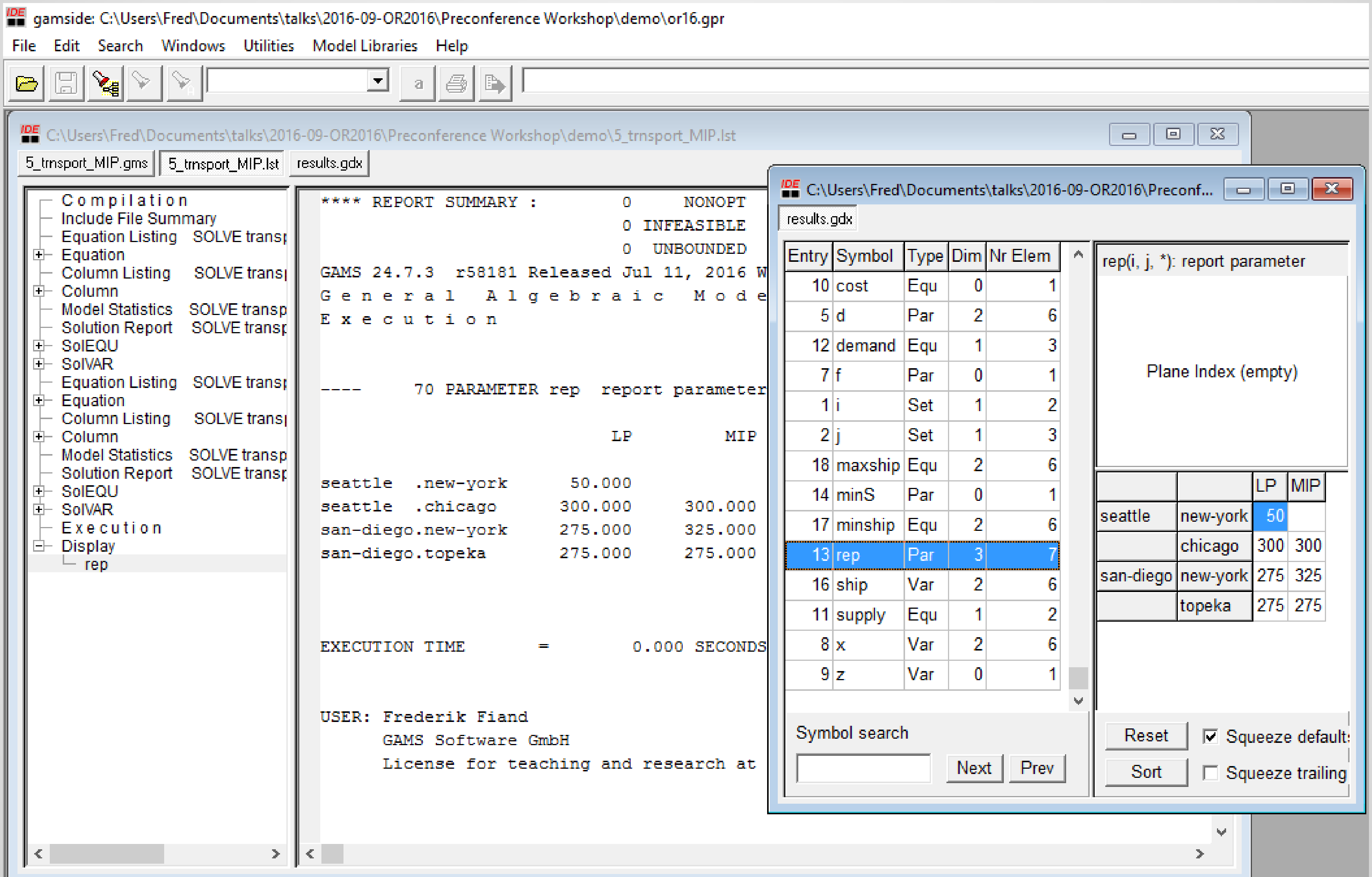
C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\5_trnsport_MIP.gms

5_trnsport_MIP.gms | 5_trnsport_MIP.lst

```
  demand(j) ..   sum(i, x(i,j))  =g=  b(j) ;

  model transportLP / all /;

  Solve transportLP using LP minimizing z ;

parameter rep(i,j,*) report parameter;
rep(i,j,'LP') = x.l(i,j);

* MIP
scalar minS minimum shipment / 100 /
       bigM big M;
bigM = min(smax(i,a(i)), smax(j,b(j)));

binary variable ship(i,j) '1 if we ship from i to j, otherwise 0';

equation minship(i,j) minimum shipment
         maxship(i,j) maximum shipment;

minship(i,j).. x(i,j) =g= minS * ship(i,j);
maxship(i,j).. x(i,j) =l= bigM * ship(i,j);

Model transportMIP / transportLP, minship, maxship / ;
option optcr = 0;

Solve transportMIP using MIP minimizing z ;

rep(i,j,'MIP') = x.l(i,j);
display rep;
```

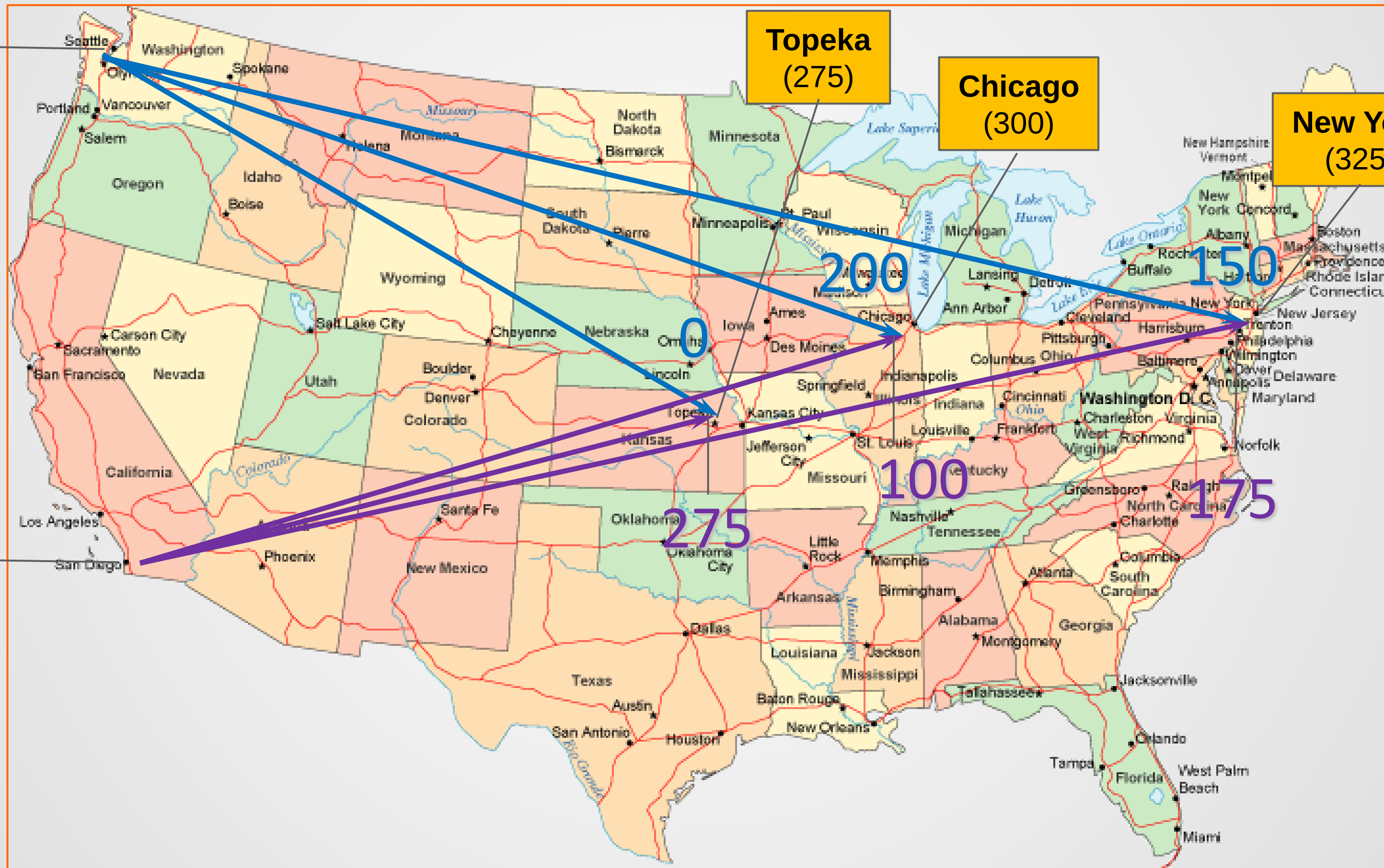
Hands-On

MIP Model: Results



MIP Model: Solution

Canning Plants (supply) —shipments (Number of cases)→ **Markets** (demand)



Freight: $90 case / thousand miles

Total cost: $153,675

Model types in this example

LP

- Determine minimum transportation cost.
  Result: city to city shipment volumes.

MIP

- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.

MINLP

- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

SP

- Allows uncertainty,
  e.g. uncertain demand

Model types in this example

LP

- Determine minimum transportation cost.
  Result: city to city shipment volumes.

MIP

- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.

MINLP

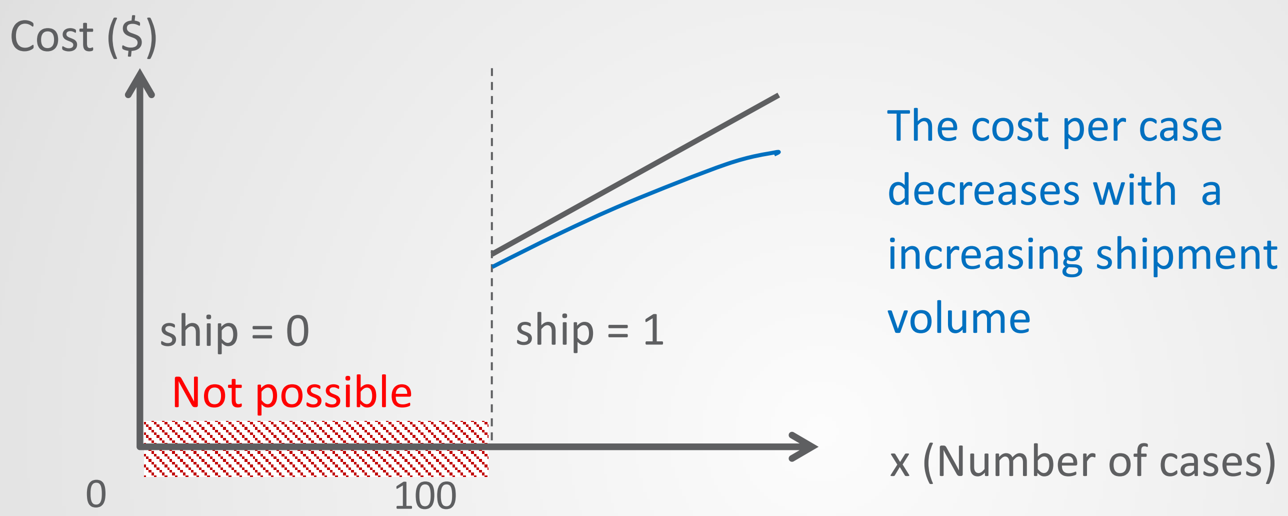
- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

SP

- Allows uncertainty,
  e.g. uncertain demand

GAMS

MINLP: Cost Savings



The cost per case decreases with a increasing shipment volume

Replace:

min $\sum_i \sum_j c_{ij} \cdot x_{ij}$ (Minimize total transportation cost)

With

min $\sum_i \sum_j c_{ij} \cdot {x_{ij}}^{\boldsymbol{beta}}$ (Minimize total transportation cost)

GAMS

MINLP Model: GAMS Syntax

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

gdx results

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\6_trnsport_MINLP.gms

6_trnsport_MINLP.gms | 6_trnsport_MINLP.lst | results.gdx

```
* MINLP
Scalar    beta / 0.95 /
Equation  costnlp define non-linear objective function;
costnlp.. z  =e=  sum((i,j), c(i,j)*x(i,j)**beta) ;

Model transportMINLP / transportMIP - cost + costnlp /;

Solve transportMINLP using MINLP minimizing z ;

rep(i,j,'MINLP') = x.l(i,j);
display rep;
```

GAMS

MINLP Model: GAMS Syntax

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

gdx results

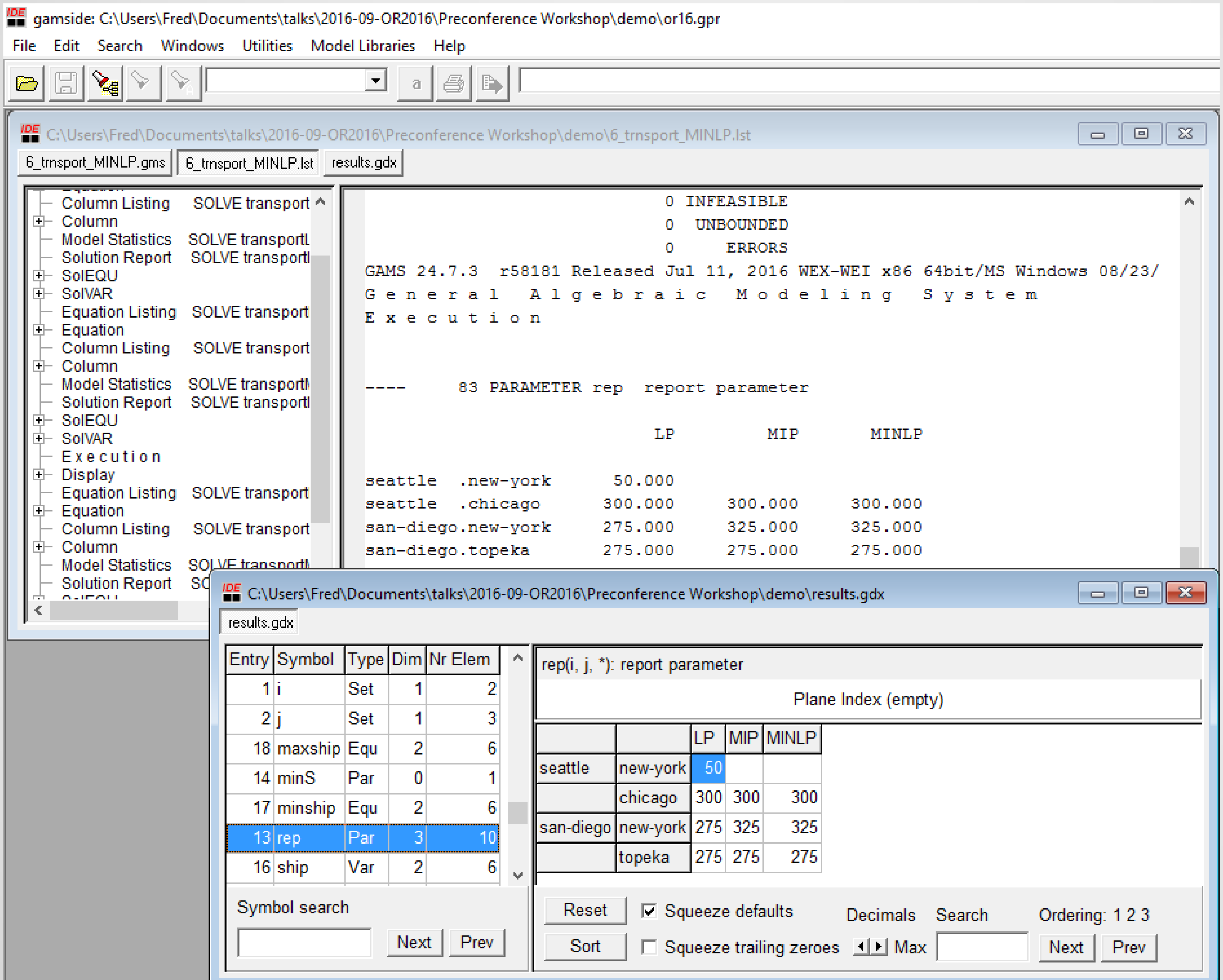
C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\6_trnsport_MINLP.gms

6_trnsport_MINLP.gms | 6_trnsport_MINLP.lst | results.gdx

```
* MINLP
Scalar    beta / 0.95 /
Equation  costnlp define non-linear objective function;
costnlp.. z  =e=  sum((i,j), c(i,j)*x(i,j)**beta) ;

Model transportMINLP / transportMIP - cost + costnlp /;

Solve transportMINLP using MINLP minimizing z ;

rep(i,j,'MINLP') = x.l(i,j);
display rep;
```

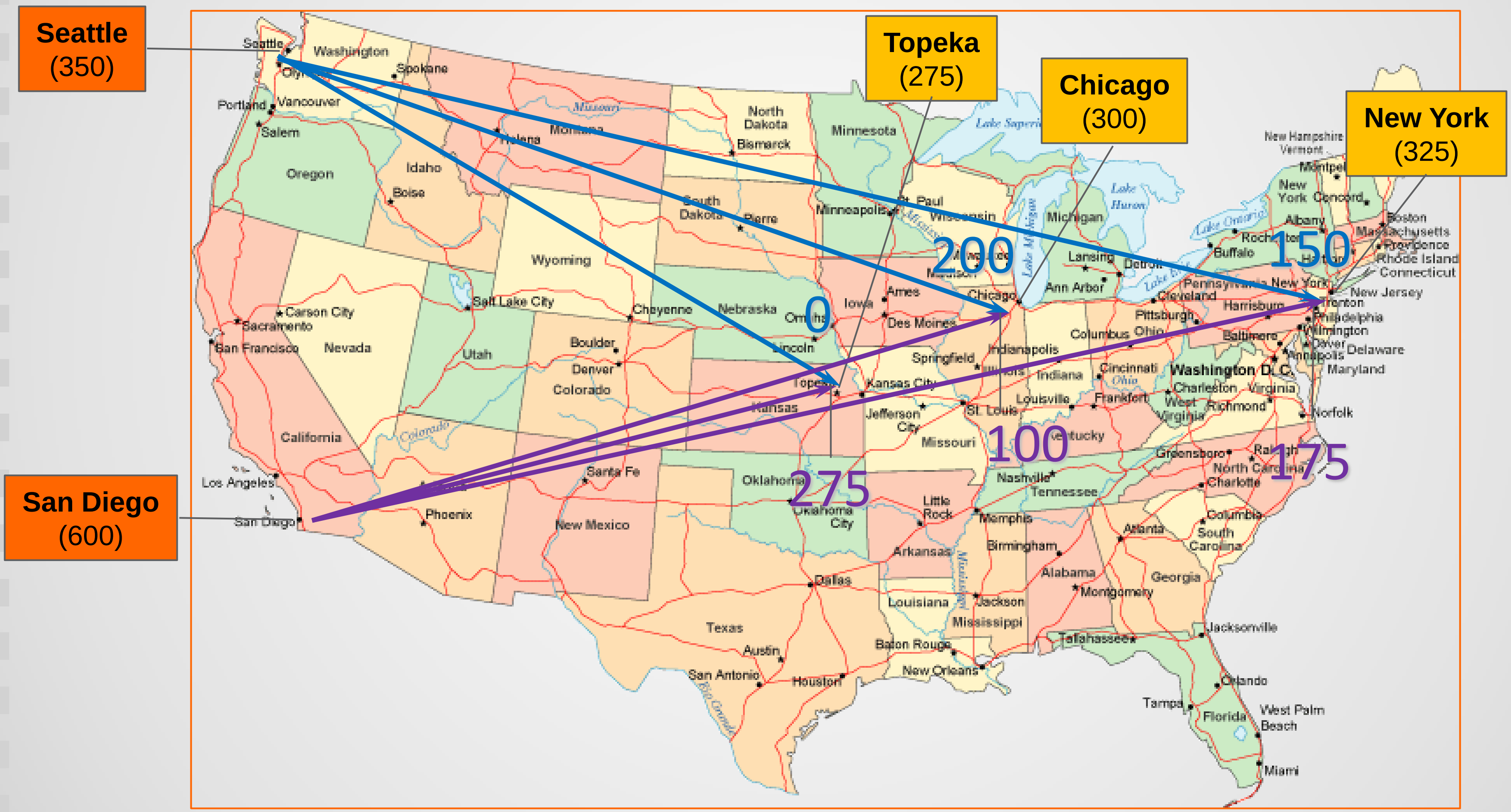
Hands-On

MINLP Model: Results



MINLP Model: Solution

Canning Plants (supply) — shipments (Number of cases) → **Markets** (demand)



Freight: $90 case / thousand miles

Total cost: $153,675

GAMS

Model types in this example

LP

- Determine minimum transportation cost.
  Result: city to city shipment volumes.

MIP

- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.

MINLP

- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

SP

- Allows uncertainty,
  e.g. uncertain demand

Model types in this example

LP

- Determine minimum transportation cost.
  Result: city to city shipment volumes.

MIP

- Allows discrete decisions,
  e.g. if we ship, then we ship at least 100 cases.

MINLP

- Allows non-linearity,
  e.g. a smooth decrease in unit cost when shipping volumes grows

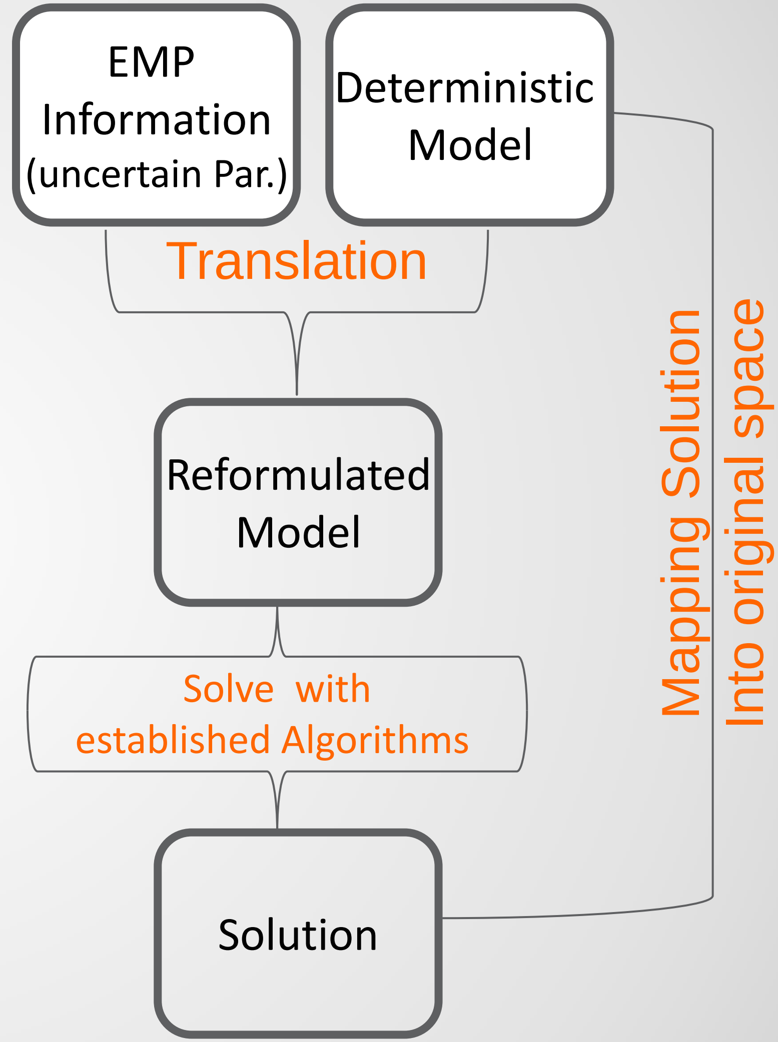
SP

- Allows uncertainty,
  e.g. uncertain demand

Stochastic Programming in GAMS

EMP/SP

- Simple interface to add uncertainty to existing deterministic models
- (EMP) Keywords to describe uncertainty include: discrete and parametric random variables, stages, chance constraints, Value at Risk, ...
- Available solution methods:
  - Automatic generation of <u>D</u>eterministic <u>E</u>quivalent (can be solved with any solver)
  - Specialized commercial algorithms (DECIS, LINDO)

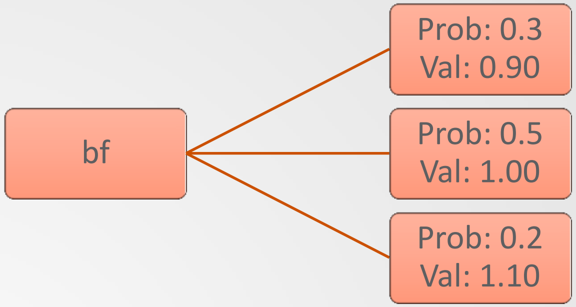


GAMS

Transport Example - Uncertain Demand

b(j): demand at market j in cases	
new-york	325
chicago	300
topeka	275

Uncertain demand factor *bf*



Decisions to make

- First-stage decision: How many units should be shipped “here and now” (without knowing the outcome)
- Second-stage (recourse) decision:
  - How can the model react if we do not ship enough?
  - Penalties for “bad” first-stage decisions, e.g. buy additional cases `u(j)` at the demand location:

```
costsp ..  z  =e=  sum((i,j), c(i,j)*x(i,j))+
sum(j,0.3*u(j));
demandsp(j) .. sum(i, x(i,j))  =g=  bf*b(j) - u(j) ;
```

GAMS

Uncertain Demand - GAMS Algebra

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

infes

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\7_trnsport_SP.gms

7_trnsport_SP.gms | 7_trnsport_SP.lst | results.gdx

```
* Stochastic Program with uncertain demand
Positive variable u(j) unsatisfied demand;
Scalar   bf demand factor / 1 /;
Equation costsp      define objective function for SP
         demandsp(j) demand satisfaction in SP;

costsp..       z =e= sum((i,j), c(i,j)*x(i,j)) + sum(j, 0.3*u(j));
demandsp(j)..  sum(i, x(i,j)) =g= bf*b(j) - u(j);

Model transportSP / costsp, demandsp, supply /;
File emp / '%emp.info%' /; put emp;
$onput
randvar bf discrete 0.3  0.9
                    0.5  1.0
                    0.2  1.1
stage 2 bf u demandsp
$offput
Putclose emp;

Set scen scenarios / s1*s4 /;
Parameter
    s_bf(scen)     demand factor for realization by scenario
    s_x(scen,i,j) shipment per scenario
    s_u(scen,j)   unsatisfied demand per scenario (bought cases);

Set dict / scen . scenario . ''
           bf   . randvar  . s_bf
           x    . level    . s_x
           u    . level    . s_u /;

option emp=lindo;
Solve transportSP min z use emp scenario dict;
```

GAMS

Uncertain Demand - GAMS Algebra

gamside: C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\or16.gpr

File Edit Search Windows Utilities Model Libraries Help

infes

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\7_trnsport_SP.gms

7_trnsport_SP.gms | 7_trnsport_SP.lst | results.gdx

```
* Stochastic Program with uncertain demand
Positive variable u(j) unsatisfied demand;
Scalar   bf demand factor / 1 /;
Equation costsp      define objective function for SP
         demandsp(j) demand satisfaction in SP;

costsp..       z =e= sum((i,j), c(i,j)*x(i,j)) + sum(j, 0.3*u(j));
demandsp(j)..  sum(i, x(i,j)) =g= bf*b(j) - u(j);

Model transportSP / costsp, demandsp, supply /;
File emp / '%emp.info%' /; put emp;
$onput
randvar bf discrete 0.3  0.9
                    0.5  1.0
                    0.2  1.1
stage 2 bf u demandsp
$offput
Putclose emp;

Set scen scenarios / s1*s4 /;
Parameter
    s_bf(scen)     demand factor for realization by scenario
    s_x(scen,i,j)  shipment per scenario
    s_u(scen,j)    unsatisfied demand per scenario (bought cases);

Set dict / scen . scenario . ''
           bf   . randvar  . s_bf
           x    . level    . s_x
           u    . level    . s_u /;

option emp=lindo;
Solve transportSP min z use emp scenario dict;
```

Hands-On

GAMS

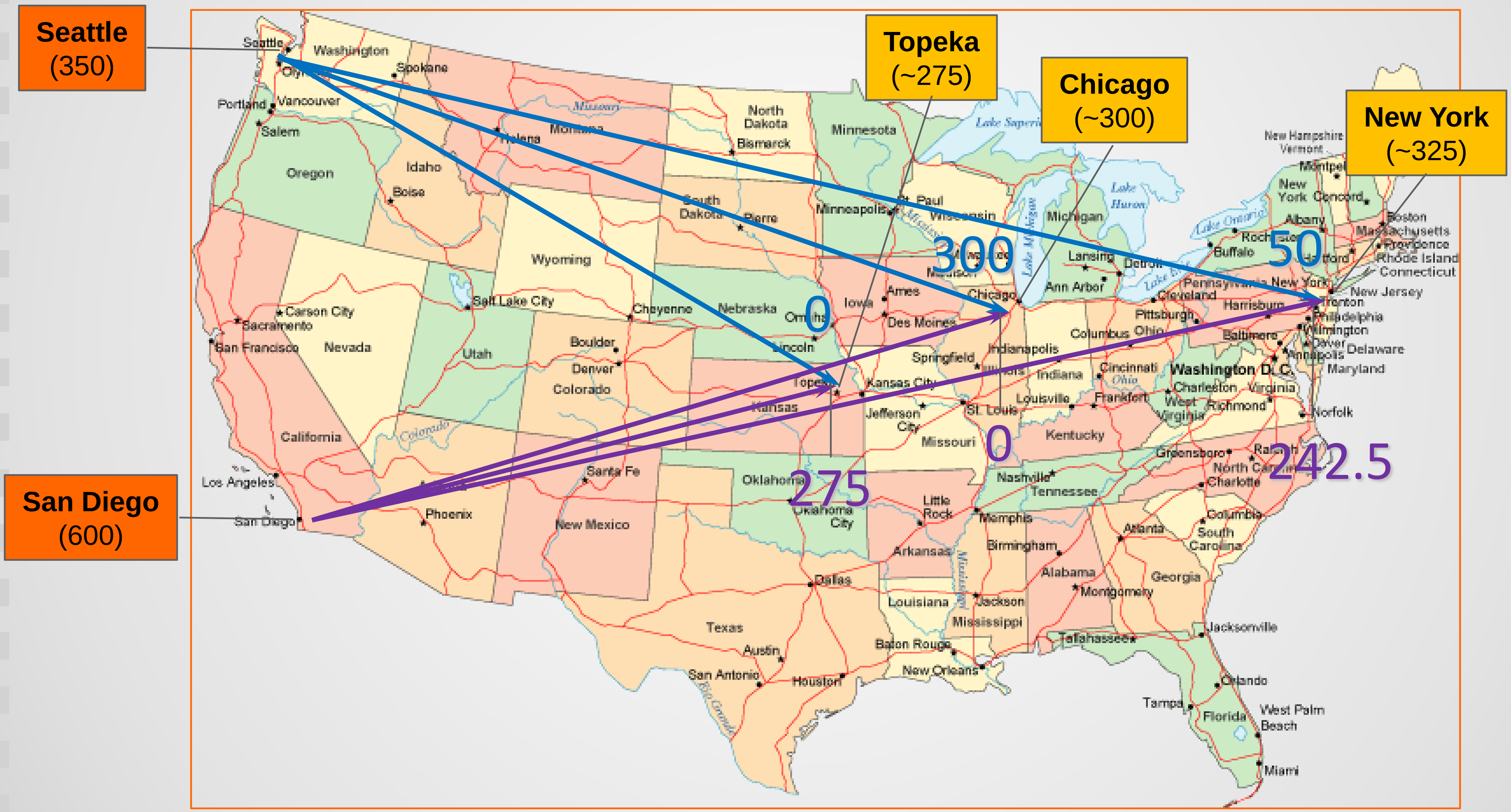
Uncertain Demand - Results

```
----     122 PARAMETER s_bf  demand factor for realization by scenario

s1 0.900,    s2 1.000,    s3 1.100


----     122 PARAMETER s_b  demand per scenario

      new-york     chicago      topeka

s1     292.500     270.000     247.500
s2     325.000     300.000     275.000
s3     357.500     330.000     302.500


----     122 PARAMETER s_x  shipment per scenario

                new-york     chicago      topeka

s1.seattle        50.000     300.000
s1.san-diego     242.500                 275.000
s2.seattle        50.000     300.000
s2.san-diego     242.500                 275.000
s3.seattle        50.000     300.000
s3.san-diego     242.500                 275.000


----     122 PARAMETER s_u  unsatisfied demand per scenario (bought cases)

      new-york     chicago      topeka

s2      32.500
s3      65.000      30.000      27.500
```

GAMS

Stochastic Program: Solution

Canning Plants (supply) — shipments (Number of cases) → **Markets** (demand)



Freight: $90 case / thousand miles

Total cost: $158,588

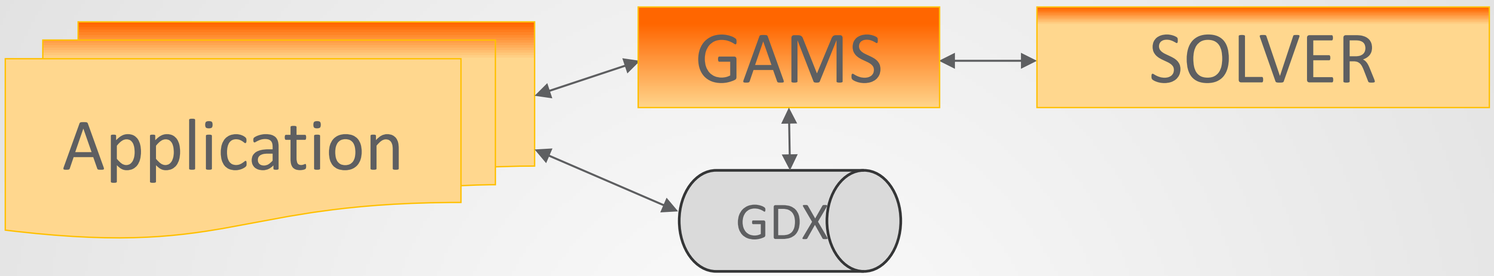
Stochastic Programming in GAMS

- The Extended Mathematical Programming (EMP) framework is used to replace parameters in the model by random variables
- Support for Multi-stage recourse problems and chance constraint models
- Easy to add uncertainty to existing deterministic models, to either use specialized algorithms or create Deterministic Equivalent (new free solver DE)
- More information: http://www.gams.com/dd/docs/solvers/empsp.pdf

Agenda

- GAMS at a Glance
- GAMS - Hands On Examples
- **APIs - Application Programming Interfaces to GAMS**
- Outlook - Some Advanced GAMS Features

Calling GAMS from your Application



Creating Input for GAMS Model

→ Data handling using **GDX** API

Callout to GAMS

→GAMS option settings using **Option** API

→Starting GAMS using **GAMS** API

Reading Solution from GAMS Model

→ Data handling using **GDX** API

Low level APIs → Object Oriented API

- Low level APIs
  - GDX, OPT, GAMSX, GMO, …
  - High performance and flexibility
  - Automatically generated imperative APIs for several languages (C, Delphi, Java, Python, C#, …)
- Object Oriented GAMS API
  - Additional layer on top of the low level APIs
  - Object Oriented
  - Written by hand to meet the specific requirements of different Object Oriented languages

GAMS

Transport Application GUI Example

- Scenario solves of the transportation problem

- Features:
  - Preparation of input data
  - Loading data from Access file
  - Solving multiple scenarios of a model
  - Displaying results

- Four implementation steps:
  1. Graphical User Interface
  2. Preparation of GAMS model
  3. Implementation of scenario solving using GAMSJob
  4. GAMSModelInstance for performance improvements

Transport Application GUI Example

- Scenario solves of the transportation problem

- Features:
  - Preparation of input data
  - Loading data from Access file
  - Solving multiple scenarios of a model
  - Displaying results

- Four implementation steps:
  1. Graphical User Interface
  2. Preparation of GAMS model
  3. Implementation of scenario solving using GAMSJob
  4. GAMSModelInstance for performance improvements

Hands-On

GAMS

Agenda

- GAMS at a Glance
- GAMS - Hands On Examples
- APIs - Application Programming Interfaces to GAMS
- **Outlook - Some Advanced GAMS Features**

Solvelink Option

controls GAMS function when linking to solve

```
Model transport /all/ ;
Option solvelink = {     %Solvelink.ChainScript%,
                         %Solvelink.CallScript%,
                         %Solvelink.CallModule%,
                         %Solvelink.AsyncGrid%,
                         %Solvelink.AsyncSimulate%,
                         %Solvelink.LoadLibrary%};
solve transport using lp minimizing z;
```

Solvelink Option

controls GAMS function when linking to solve

```
Model transport /all/ ;
Option solvelink = {     %Solvelink.ChainScript%,
                         %Solvelink.CallScript%,
                         %Solvelink.CallModule%,
                         %Solvelink.AsyncGrid%,
                         %Solvelink.AsyncSimulate%,
                         %Solvelink.LoadLibrary%};
solve transport using lp minimizing z;
```

- ChainScript [0]: Solver process, GAMS vacates memory

  + Maximum memory available to solver

  + protection against solver failure (*hostile* link)

  - swap to disk

GAMS

Solvelink Option – cont.

- Call{Script [1]/Module [2]}: Solver process, GAMS stays live
  - + protection against solver failure (*hostile* link)
  - + no swap of GAMS database
  - - file based model communication

Solvelink Option – cont.

- Call{Script [1]/Module [2]}: Solver process, GAMS stays live
  - + protection against solver failure (*hostile* link)
  - + no swap of GAMS database
  - - file based model communication

- LoadLibrary [5]: Solver DLL in GAMS process
  - + fast memory based model communication
  - + update of model object inside the solver (hot start)
  - - not supported by all solvers

GAMS

Simple Serial Solve - Performance

trnsport.gms (LP) solved 500 times with CPLEX:

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\trnsport_GUSS_solvelin...

trnsport_GUSS_solvelink.gms

```
loop(s,
  d(i,j) = dd(s,i,j);
  f      = ff(s);
  Solve transport using lp minimizing z ;
  rep(s) = z.l;
);
```

GAMS

Simple Serial Solve - Performance

trnsport.gms (LP) solved 500 times with CPLEX:

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo\trnsport_GUSS_solvelin...

trnsport_GUSS_solvelink.gms

```
loop(s,
  d(i,j) = dd(s,i,j);
  f      = ff(s);
  Solve transport using lp minimizing z ;
  rep(s) = z.l;
);
```

The **solvelink** option controls GAMS function when linking to solve.

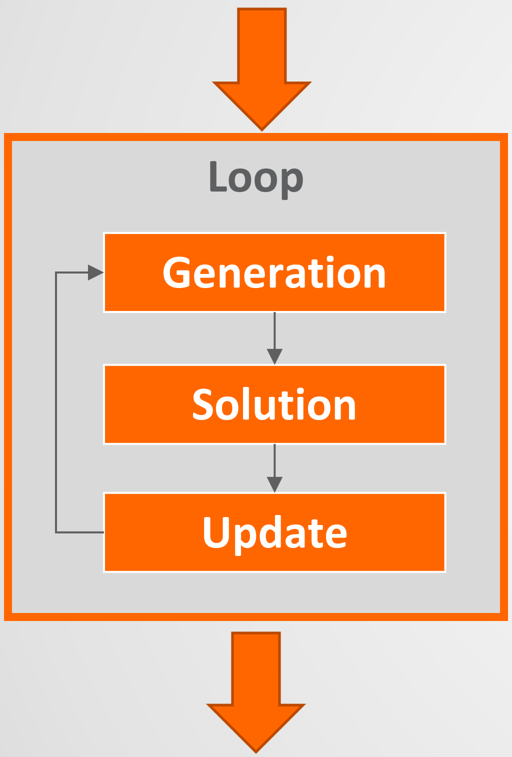
Setting	Solve time (secs)
Solvelink=%Solvelink.ChainScript%	54.368
Solvelink=%Solvelink.CallModule%	42.909
Solvelink=%Solvelink.LoadLibrary%	05.039

Hands-On

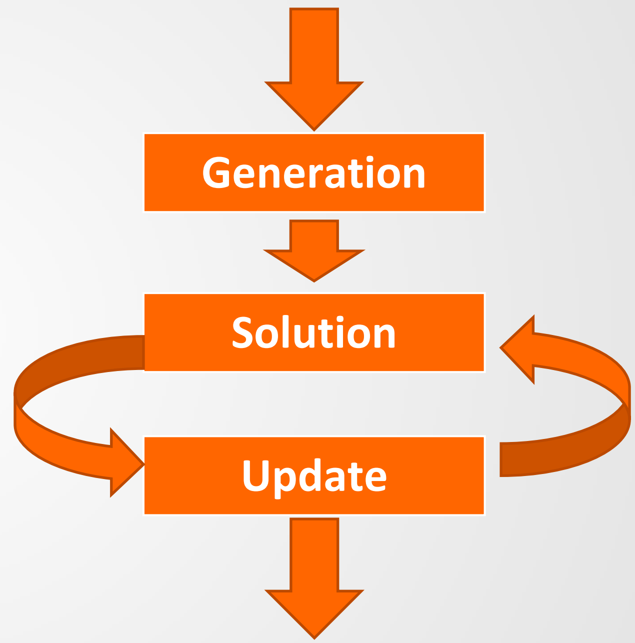
GAMS

Scenario Solver

Simple Serial Solve Loop



Scenario Solver/GUSS



Generates model once and updates the algebraic model **keeping the model "hot"** inside the solver.

GAMS

Scenario Solver/GUSS - Performance

trnsport.gms (LP) solved 500 times with CPLEX:

```
* GUSS
tmp = jnow;
Set mattrib / system.GUSSModelAttributes /;
Parameter
    xxGUSS(s,i,j)     collector for level of x
    srep(s, mattrib) model attibutes like modelstat etc
    o(*)              GUSS options /  SkipBaseCase 1 /

Set dict    / s . scenario.''
              o . opt      .srep
              d . param    .dd
              f . param    .ff
              x . level    .xxGUSS /

Solve transport using lp minimizing z scenario dict;
```

time(*): time for 500 scenarios	
ChainScript	54.368
CallModule	12.909
LoadLibrary	5.039
GUSS	1.947

Setting	Solve time (secs)
Solvelink=%Solvelink.ChainScript%	54.368
Solvelink=%Solvelink.CallModule%	12.909
Solvelink=%Solvelink.LoadLibrary%	05.039
GUSS	01.947

GAMS

Scenario Solver/GUSS - Performance

Example: Stochastic model with 66,320 linear problems

Setting	Solve time (secs)
Loop: Solvelink=%Solvelink.Chainscript (default)	7,204
Loop: Solvelink=%Solvelink.LoadLibrary%	2,481
GAMS Scenario Solver	392
CPLEX Concert Technology	210

Factor

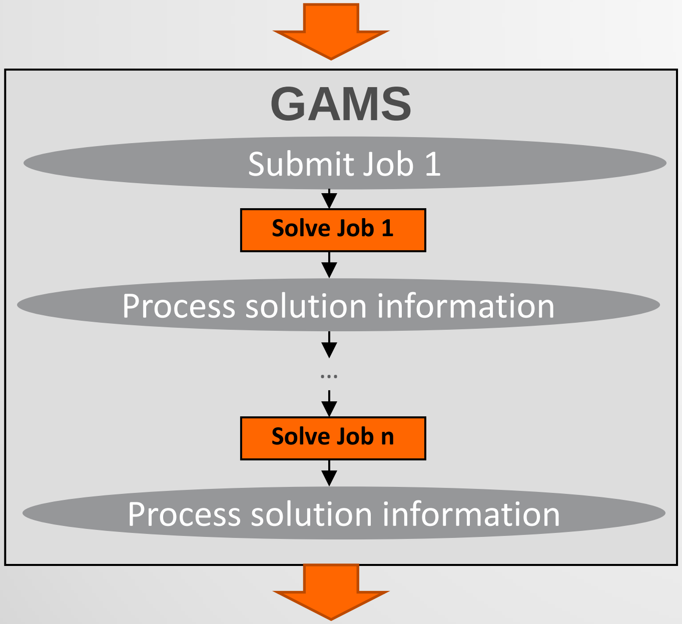
18.3

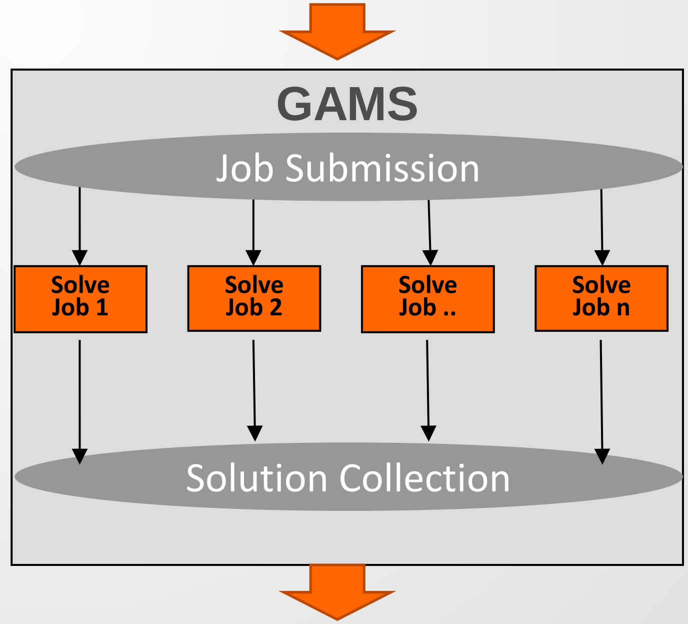
1.86

Grid Computing Facility

GAMS jobs in a **distributed** environment

- Scalable: supports large grids, but also works on local machine
- Platform independent, works with all solvers/model types
- Only minor changes to model required





GAMS

Grid Computing Facility – Example 1

➢ GAMS Model Library: trnsgrid

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\dem

9a_trnsgrid.gms 9a_trnsgrid.lst

```
transport.solvelink = %solvelink.AsyncGrid%;  // tur
transport.limcol    = 0;
transport.limrow    = 0;
transport.solprint  = %solprint.Quiet%;

set s scenarios / 1*5 /;

parameter dem(s,j)  random demand
          h(s) store the instance handle;

dem(s,j)= b(j)*uniform(.95,1.15); // create some random demands

loop(s,
   b(j) = dem(s,j)
   Solve transport using lp minimizing z;
   h(s) = transport.handle );     // save instance handle

parameter repx(s,i,j) solution report
          repy         summary report;

repy(s,'solvestat') = na;
repy(s,'modelstat') = na;

* we use the handle parameter to indicate that the solution has been collected
repeat
   loop(s$handlecollect(h(s)),
      repx(s,i,j) = x.l(i,j);
      repy(s,'solvestat') = transport.solvestat;
      repy(s,'modelstat') = transport.modelstat;
      repy(s,'resusd'   ) = transport.resusd;
      repy(s,'objval')    = transport.objval;
      display$handledelete(h(s)) 'trouble deleting handles' ;
      h(s) = 0 ) ;    // indicate that we have loaded the solution
   display$sleep(card(h)*0.2) 'was sleeping for some time';
until card(h) = 0 or timeelapsed > 10;  // wait until all models are loaded

display repx, repy;

abort$sum(s$(repy(s,'solvestat')=na),1) 'Some jobs did not return';
```

```
          GAMS Software GmbH
          License for teaching and research at degree granting i
--- Starting compilation
--- 9a_trnsgrid.gms(101) 3 Mb
--- Starting execution: elapsed 0:00:00.003
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
--- LOOPS s = 1
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000001
--- Executing after solve: elapsed 0:00:00.046
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
--- LOOPS s = 2
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000002
--- Executing after solve: elapsed 0:00:00.066
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
--- LOOPS s = 3
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000003
--- Executing after solve: elapsed 0:00:00.091
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
--- LOOPS s = 4
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000004
--- Executing after solve: elapsed 0:00:00.121
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
```

Close Open Log ☐ Summary only ☑ Update

Grid Computing Facility – Example 1

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\dem

9a_trnsgrid.gms | 9a_trnsgrid.lst

➢ GAMS Model Library: trnsgrid

```
transport.solvelink = %solvelink.AsyncGrid%;  // tur
transport.limcol    = 0;
transport.limrow    = 0;
transport.solprint  = %solprint.Quiet%;

set s scenarios / 1*5 /;

parameter dem(s,j)  random demand
          h(s)  store the instance handle;

dem(s,j)= b(j)*uniform(.95,1.15); // create some random demands

loop(s,
   b(j) = dem(s,j)
   Solve transport using lp minimizing z;
   h(s) = transport.handle );     // save instance handle

parameter repx(s,i,j) solution report
          repy         summary report;

repy(s,'solvestat') = na;
repy(s,'modelstat') = na;

* we use the handle parameter to indicate that the solution has been collected
repeat
   loop(s$handlecollect(h(s)),
      repx(s,i,j) = x.l(i,j);
      repy(s,'solvestat') = transport.solvestat;
      repy(s,'modelstat') = transport.modelstat;
      repy(s,'resusd'   ) = transport.resusd;
      repy(s,'objval')    = transport.objval;
      display$handledelete(h(s)) 'trouble deleting handles' ;
      h(s) = 0 ) ;    // indicate that we have loaded the solution
   display$sleep(card(h)*0.2) 'was sleeping for some time';
until card(h) = 0 or timeelapsed > 10;  // wait until all models are loaded

display repx, repy;

abort$sum(s$(repy(s,'solvestat')=na),1) 'Some jobs did not return';
```

```
          GAMS Software GmbH
          License for teaching and research at degree granting i
--- Starting compilation
--- 9a_trnsgrid.gms(101) 3 Mb
--- Starting execution: elapsed 0:00:00.003
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
--- LOOPS s = 1
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000001
--- Executing after solve: elapsed 0:00:00.046
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
--- LOOPS s = 2
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000002
--- Executing after solve: elapsed 0:00:00.066
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
--- LOOPS s = 3
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000003
--- Executing after solve: elapsed 0:00:00.091
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
--- LOOPS s = 4
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000004
--- Executing after solve: elapsed 0:00:00.121
--- 9a_trnsgrid.gms(75) 4 Mb
--- Generating LP model transport
--- 9a_trnsgrid.gms(77) 4 Mb
```

Close | Open Log | ☐ Summary only | ☑ Update

Hands-On

Grid Computing Facility – Example 2

➢ GAMS Model Library: tgridmix

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo

tgridmix.gms | tgridmix.lst

```
dem(s,j)= b(j)*uniform(.95,1.15); // create some random demands
loop(sl,
  tStart = jnow;
  repy(sl,s,'solvestat') = na;
  repy(sl,s,'modelstat') = na;
  actS(s) = no; h(s) = 0; nexS(s) = sameas('1',s);
  transport.solvelink = slnum(sl);

  repeat
     while (card(actS)<maxS and card(nexS),
        loop(nexS(s),
           b(j) = dem(s,j)
           Solve transport using lp minimizing z;
           h(s) = transport.handle;
           actS(s) = yes;
        );
        nexS(s) = nexS(s-1); // advance nexS
     );
     colS(s) = no;
     display$ReadyCollect(h) 'Waiting for next instance to collect';
     loop(actS(s)$handlecollect(h(s)),
        repx(sl,s,i,j) = x.l(i,j);
        repy(sl,s,'solvestat') = transport.solvestat;
        repy(sl,s,'modelstat') = transport.modelstat;
        repy(sl,s,'resusd'   ) = transport.resusd;
        repy(sl,s,'objval')    = transport.objval;
        display$handledelete(h(s)) 'trouble deleting handles' ;
        colS(s) = yes; h(s) = 0;
     ); actS(colS) = no;
  until (card(nexS)=0 and card(actS) = 0) or timeelapsed > 10;  // wait until a
  repy(sl,'time','elapsed') = (jnow - tStart)*3600*24;
  abort$sum(s$(repy(sl,s,'solvestat')=na),1) 'Some jobs did not return';
);
display repx, repy;
```

```
---         FOR/WHILE = 2
---         FOR/WHILE = 1
---        * = 8
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000018
--- Executing after solve: elapsed 0:00:00.742
--- tgridmix.gms(89) 4 Mb
--- Generating LP model transport
--- tgridmix.gms(100) 4 Mb
--- LOOPS sl = Grid
---         FOR/WHILE = 2
---         FOR/WHILE = 2
---        * = 9
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000019
--- Executing after solve: elapsed 0:00:00.759
--- tgridmix.gms(89) 4 Mb
--- Generating LP model transport
--- tgridmix.gms(100) 4 Mb
--- LOOPS sl = Grid
---         FOR/WHILE = 2
---         FOR/WHILE = 3
---        * = 10
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000020
--- Executing after solve: elapsed 0:00:00.776
--- tgridmix.gms(89) 4 Mb
--- GDXin=C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconfer
--- Removed handle grid145000018
--- GDXin=C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconfer
--- Removed handle grid145000019
--- GDXin=C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconfer
--- Removed handle grid145000020
--- tgridmix.gms(121) 4 Mb
*** Status: Normal completion
--- Job tgridmix.gms Stop 08/25/16 06:27:43 elapsed 0:00:00.849
```

Close | Open Log | ☐ Summary only | ☑ Update

Grid Computing Facility – Example 2

➤ GAMS Model Library: tgridmix

C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconference Workshop\demo

tgridmix.gms | tgridmix.lst

```
dem(s,j)= b(j)*uniform(.95,1.15); // create some random demands
loop(sl,
  tStart = jnow;
  repy(sl,s,'solvestat') = na;
  repy(sl,s,'modelstat') = na;
  actS(s) = no; h(s) = 0; nexS(s) = sameas('1',s);
  transport.solvelink = slnum(sl);

  repeat
     while (card(actS)<maxS and card(nexS),
        loop(nexS(s),
           b(j) = dem(s,j)
           Solve transport using lp minimizing z;
           h(s) = transport.handle;
           actS(s) = yes;
        );
        nexS(s) = nexS(s-1); // advance nexS
     );
     colS(s) = no;
     display$ReadyCollect(h) 'Waiting for next instance to collect';
     loop(actS(s)$handlecollect(h(s)),
        repx(sl,s,i,j) = x.l(i,j);
        repy(sl,s,'solvestat') = transport.solvestat;
        repy(sl,s,'modelstat') = transport.modelstat;
        repy(sl,s,'resusd'   ) = transport.resusd;
        repy(sl,s,'objval')    = transport.objval;
        display$handledelete(h(s)) 'trouble deleting handles' ;
        colS(s) = yes; h(s) = 0;
     ); actS(colS) = no;
  until (card(nexS)=0 and card(actS) = 0) or timeelapsed > 10;  // wait until a
  repy(sl,'time','elapsed') = (jnow - tStart)*3600*24;
  abort$sum(s$(repy(sl,s,'solvestat')=na),1) 'Some jobs did not return';
);
display repx, repy;
```

```
---         FOR/WHILE = 2
---         FOR/WHILE = 1
---        * = 8
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000018
--- Executing after solve: elapsed 0:00:00.742
--- tgridmix.gms(89) 4 Mb
--- Generating LP model transport
--- tgridmix.gms(100) 4 Mb
--- LOOPS sl = Grid
---         FOR/WHILE = 2
---         FOR/WHILE = 2
---        * = 9
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000019
--- Executing after solve: elapsed 0:00:00.759
--- tgridmix.gms(89) 4 Mb
--- Generating LP model transport
--- tgridmix.gms(100) 4 Mb
--- LOOPS sl = Grid
---         FOR/WHILE = 2
---         FOR/WHILE = 3
---        * = 10
---   6 rows  7 columns  19 non-zeroes
--- Submitting model transport with handle grid145000020
--- Executing after solve: elapsed 0:00:00.776
--- tgridmix.gms(89) 4 Mb
--- GDXin=C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconfer
--- Removed handle grid145000018
--- GDXin=C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconfer
--- Removed handle grid145000019
--- GDXin=C:\Users\Fred\Documents\talks\2016-09-OR2016\Preconfer
--- Removed handle grid145000020
--- tgridmix.gms(121) 4 Mb
*** Status: Normal completion
--- Job tgridmix.gms Stop 08/25/16 06:27:43 elapsed 0:00:00.849
```

Close | Open Log | ☐ Summary only | ☑ Update

Hands-On

GAMS

Solving "many" Scenarios
How to find the right approach?

1. Small Ratio of solver time / GAMS time → Scenario Solver
2. Large ratio i.e. only solver time is relevant (pre/post processing not critical) → Grid Computing Facility
3. Entire model run including pre processing / optimization / post processing is costly → Parallel execution of entire model in the cloud

GAMS

Application - Scenario Solver

➢ Scenario Solver and Parallel Combined

Implementation	Number of MIP models	Solve time	Rest of algorithm	Total time
Traditional GAMS loop	100,000	1068 sec	169 sec	1237 sec
Scenario Solver	100,000	293 sec	166 sec	459 sec

Implementation	Number of MIP models	Worker Threads	Parallel sub-problem time	Rest of algorithm (serial)	Total time
Parallel + Scenario Solver	100,000	4	116 sec	67 sec	183 sec

http://yetanothermathprogrammingconsultant.blogspot.de/2012/04/parallel-gams-jobs-2.html

Where to Find Help?

- Documentation Center: http://gams.com/help/index.jsp
- Support Wiki: http://support.gams.com/
- Mailing List(s): http://gams.com/maillist/index.htm
- YouTube Channel: https://www.youtube.com/user/GAMSLessons
- GAMS support: support@gams.com

Meet us at the GAMS booth!

Other GAMS Talks

BEAM-ME: Acceleration Strategies for Energy System Models

Given by: Frederik Fiand
When: Thursday (Sep. 01), 09:00-09:30
Where: Hörsaal 3
Abstract: BEAM-ME is a project funded by the German Federal Ministry for Economic Affairs and Energy (BMWi) and addresses the need for new and improved solution approaches for energy system models. The project unites various partners with complementary expertise from the fields of algorithms, computing and application development. The con- sidered problems result in large-scale LPs that are computationally in- tractable for state-of-the-art solvers. Hence, new solution approaches combining decomposition methods, algorithm development and high performance computing are developed. We provide an overview on the large variety of challenges we are facing within this project, present current solutions approaches and provide first results.

Recent Enhancements in GAMS

Given by: Franz Nelissen
When: Thursday (Sep. 01), 11:30-12:00
Where: Seminarraum 105
Abstract: Algebraic Modeling Languages (AML) are one of the success stories in Operations Research. GAMS is one of the prominent AMLs and has evolved continuously in response to user requirements, changes in computing environments and advances in the theory and practice of mathematical programing. In this talk we will begin with some fundamental principles and outline several recent enhancements of GAMS supporting efficient and productive development of optimization based decision support applications.

Thank You

Europe
GAMS Software GmbH
P.O. Box 40 59
50216 Frechen, Germany
Phone: +49 221 949 9170
Fax: +49 221 949 9171
info@gams.de

USA
GAMS Development Corp.
1217 Potomac Street, NW
Washington, DC 20007, USA
Phone: +1 202 342 0180
Fax: +1 202 342 0181
sales@gams.com